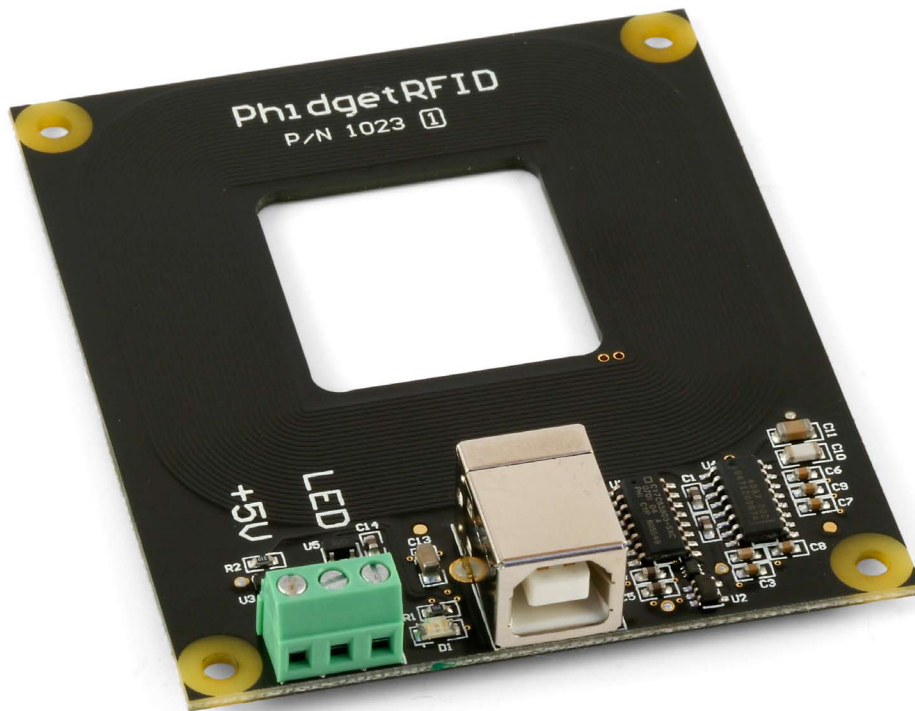


1023 - PhidgetRFID



Product Features

- Reads tags brought within 3 inches of the reader
- Reads any tag with EM4102 protocol
- Returns the unique number contained in the tag
- Provides 2 Digital Outputs to drive LEDs, relays, etc.
- On-Board LED
- Connects directly to a computer's USB port

Programming Environment

Operating Systems: Windows 2000/XP/Vista, Windows CE, Linux, and Mac OS X

Programming Languages (APIs): VB6, VB.NET, C#.NET, C++, Flash 9, Flex, Java, LabVIEW, Python, Max/MSP, and Cocoa.

Examples: Many example applications for all the operating systems and development environments above are available for download at www.phidgets.com.

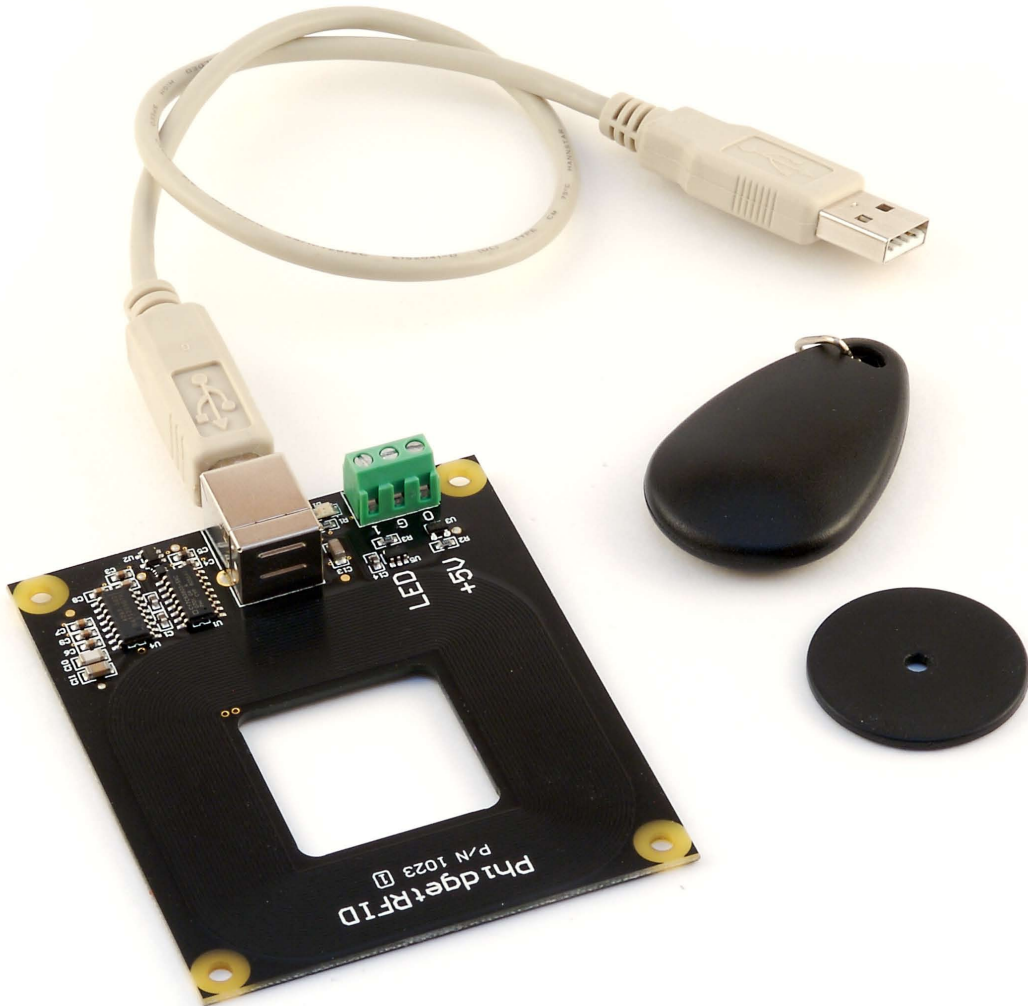
Installing the hardware

The kit contains:

- A PhidgetRFID
- A USB Cable

You will also need:

- A compatible RFID tag




Connect the PhidgetRFID board to the computer using the USB cable.

Downloading and Installing the software

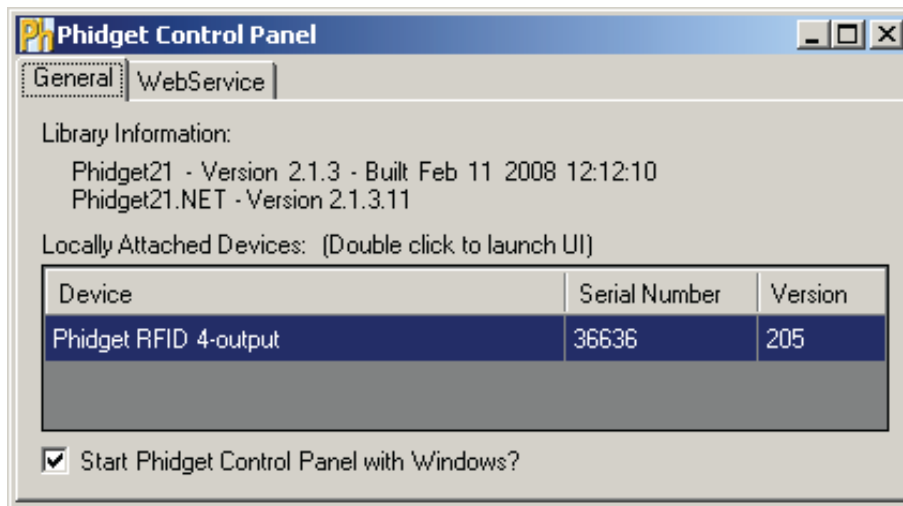
If you are using Windows 2000/XP/Vista


Go to www.phidgets.com >> Downloads >> Windows

Download and run Phidget21.MSI

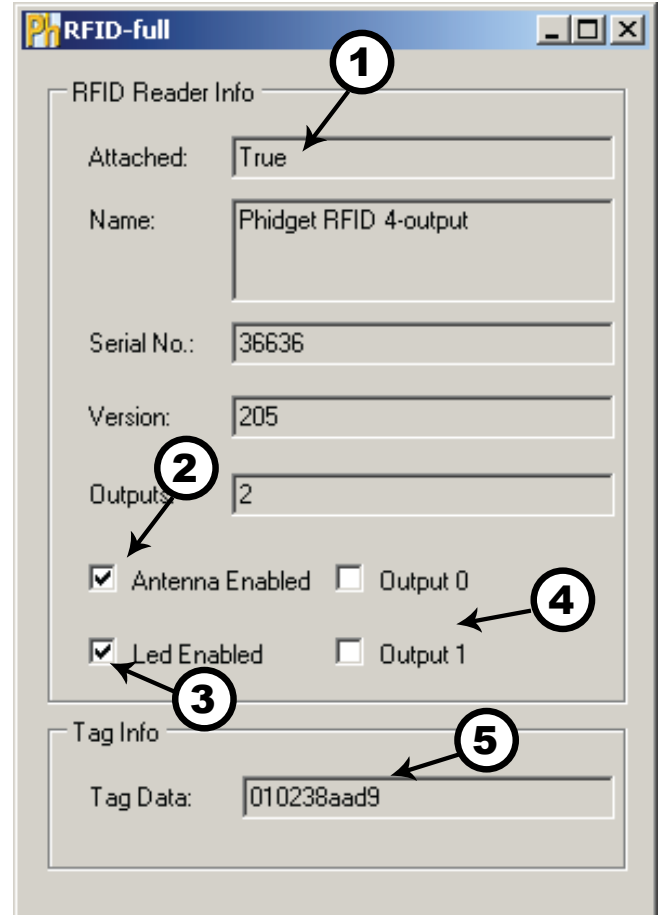
You should see the  icon on the right hand corner of the Task Bar.

Testing the PhidgetRFID Functionality



Double Click on the  icon to activate the Phidget Control Panel and make sure that **PhidgetRFID** is properly attached to your PC.

1. Double Click on **PhidgetRFID** in the Phidget Control Panel to bring up PhidgetRFID-full and check that the box labelled Attached contains the word True.
2. Click on the Antenna Enabled box to enable the antenna.
3. Click on the LED Enabled box to turn the LED on.
4. The Output 0 box controls the +5V digital output and the Output 1 box controls the external LED digital output.
5. Bring an RFID tag close to the PhidgetRFID board and check that the identification string is displayed. Make sure that the Antenna is enabled for this step.

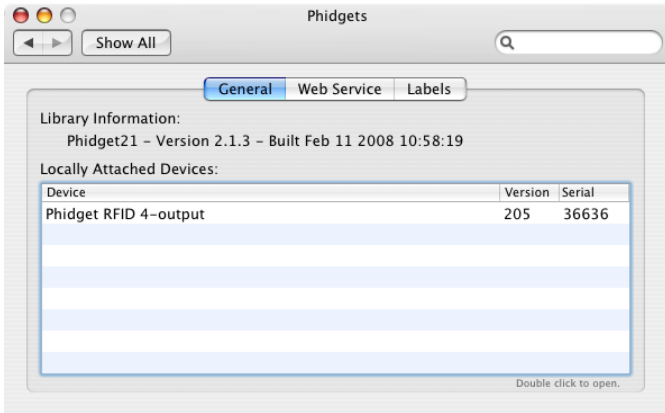


If you are using Mac OS X

Go to www.phidgets.com >> downloads >> Mac

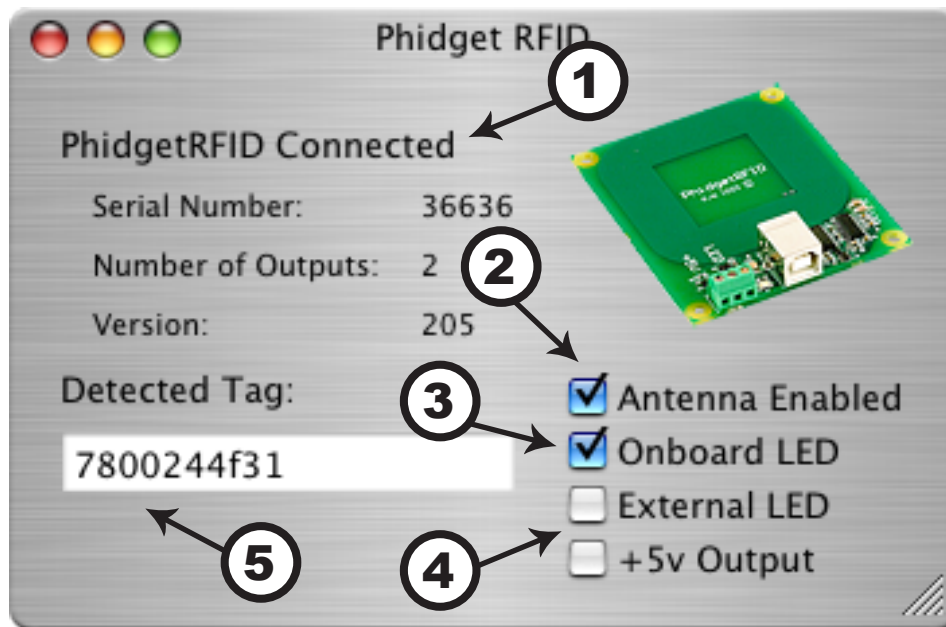
Download Mac OS X Framework

Testing the PhidgetRFID functionality



Click on System Preferences >> Phidgets (under Other) to activate the Preference Pane. Make sure that the **PhidgetRFID** is properly attached.

1. Double Click on **PhidgetRFID** in the Phidget Preference Pane to bring up PhidgetRFID Example and check that the PhidgetRFID is properly connected.



2. Click on the antenna enabled box to enable the antenna.
3. Click on the Onboard LED box to turn the LED on.
4. The next 2 boxes control the +5V digital output and an external LED (using a digital output).
5. Bring the RFID tag close to the PhidgetRFID board and check that the identification string is displayed. Make sure that the antenna is enabled.

If you are using Linux

Go to www.phidgets.com >> Downloads >> Linux

- Download Linux Source
- Have a look at the readme file
- Build Phidget21

The most popular programming languages in Linux are C/C++ and Java.

Note: Many Linux systems are now built with unsupported third party drivers. It may be necessary to uninstall these drivers for our libraries to work properly.

Note: Phidget21 for Linux is a user-space library. Applications typically have to be run as root, or udev/hotplug must be configured to give permissions when the Phidget is plugged in.

If you are using Windows Mobile/CE 5.0 or 6.0

Go to www.phidgets.com >> Downloads >> Windows Mobile/CE

Download x86 or ARMV4I, depending on the platform you are using. Mini-itx and ICOP systems will be x86, and most mobile devices, including XScale based systems will run the ARMV4I.

The CE libraries are distributed in .CAB format. Windows Mobile/CE is able to directly install .CAB files.

The most popular languages are C/C++, .NET Compact Framework (VB.NET and C#). A desktop version of Visual Studio can usually be configured to target your Windows Mobile Platform, whether you are compiling to machine code or the .NET Compact Framework.

Programming a Phidget

Phidgets' philosophy is that you do not have to be an electrical engineer in order to do projects that use devices like sensors, motors, motor controllers, and interface boards. All you need to know is how to program. We have developed a complete set of Application Programming Interfaces (API) that are supported for Windows, Mac OS X, and Linux. When it comes to languages, we support VB6, VB.NET, C#.NET, C, C++, Flash 9, Flex, Java, LabVIEW, Python, Max/MSP, and Cocoa.

Architecture

We have designed our libraries to give you the maximum amount of freedom. We do not impose our own programming model on you.

To achieve this goal we have implemented the libraries as a series of layers with the C API at the core surrounded by other language wrappers.

Libraries

The lowest level library is the C API. The C API can be programmed against on Windows, CE, OS X and Linux. With the C API, C/C++, you can write cross-platform code. For systems with minimal resources (small computers), the C API may be the only choice.

The Java API is built into the C API Library. Java, by default is cross-platform - but your particular platform may not support it (CE).

The .NET API also relies on the C API. Our default .NET API is for .NET 2.0 Framework, but we also have .NET libraries for .NET 1.1 and .NET Compact Framework (CE).

The COM API relies on the C API. The COM API is programmed against when coding in VB6, VBScript, Excel (VBA), Delphi and Labview.

The ActionScript 3.0 Library relies on a communication link with a PhidgetWebService (see below). ActionScript 3.0 is used in Flex and Flash 9.

Programming Hints

- Every phidget has a unique serial number - this allows you to sort out which device is which at runtime. Unlike USB devices which model themselves as a COM port, you don't have to worry about where in the USB bus you plug your phidget in. If you have more than one phidget, even of the same type, their serial numbers enable you to sort them out at runtime.
- Each phidget you have plugged in is controlled from your application using an object/handle specific to that phidget. This link between the phidget and the software object is created when you call the .OPEN group of commands. This association will stay, even if the phidget is disconnected/reattached, until .CLOSE is called.
- The Phidget APIs are designed to be used in an event-driven architecture. While it is possible to poll them, we don't recommend it. Please familiarize yourself with event programming.

Networking Phidgets

The PhidgetWebService is an application written by Phidgets Inc. which acts as a network proxy on a computer. The PhidgetWebService will allow other computers on the network to communicate with the Phidgets connected to that computer. ALL of our APIs

have the capability to communicate with Phidgets on another computers that has the PhidgetWebService running.

The PhidgetWebService also makes it possible to communicate with other applications that you wrote and that are connected to the PhidgetWebService, through the PhidgetDictionary object.

API documentation

We maintain API manuals for COM (Windows), C (Windows/Mac OSX/Linux), Action Script, .Net and Java. Look at the section that corresponds to the Phidget you are using. These manuals can be accessed in different ways:

Using Downloads on main menu

Click on Downloads >> Operating System (i.e. Windows) >> Platform (i.e. C#) >> API Document (i.e. Net API Manual)

Using Products on Home Page

Click on InterfaceKits (under Products) >> 1018 PhidgetInterfaceKit 8/8/8 >> API Manual (Under Software Information)

Using Information on Home Page

Click on Information (under Main Menu) >> Your API Manual (under Phidgets API Manuals)

Examples

We have written examples to illustrate how the APIs are used. Examples for the C#.NET programming language Include .exe files for each of the examples in the directory root.

Due to the large number of languages and devices we support, we cannot provide examples in every language for every phidget. Some of the examples are very minimal, and other examples will have a full-featured GUI allowing all the functionality of the device to be explored. Most developers start by modifying existing examples until they have an understanding of the architecture.

To get the examples, go to www.phidgets.com and click on Downloads. Under Step 2: click on your platform of choice and click on the File Name besides Examples.

Support

- Click on Live Support on www.phidgets.com to chat with our support desk experts
- Call the support desk at 1.403.282.7335 8:00 AM to 5:00 PM Mountain Time (US & Canada) - GMT-07:00
- E-mail sales@phidgets.com

Simple example written in C#

```
/* - RFID simple -
*****
* This program simply displays the data that is generated by an RFID phidget in a very
* simple case and outputs it to the console. This simple example covers the basics of
* connecting and using an RFID phidget. For a more detailed example, see RFID-full.
*
* Please note that this example was designed to work with only one Phidget RFID
* connected. For an example using multiple Phidget RFIDs, please see a "multiple"
* example in the RFID Examples folder.
*
* Copyright 2007 Phidgets Inc.
* This work is licensed under the Creative Commons Attribution 2.5 Canada License.
* To view a copy of this license, visit http://creativecommons.org/licenses/by/2.5/ca/
*/
using System;
using System.Collections.Generic;
using System.Text;
using Phidgets; //Needed for the RFID class and the PhidgetException class
using Phidgets.Events; //Needed for the phidget event handling classes

namespace RFID_simple
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                RFID rfid = new RFID(); //Declare an RFID object

                //initialize our Phidgets RFID reader and hook the event handlers
                rfid.Attach += new AttachEventHandler(rfid_Attach);
                rfid.Detach += new DetachEventHandler(rfid_Detach);
                rfid.Error += new ErrorEventHandler(rfid_Error);

                rfid.Tag += new TagEventHandler(rfid_Tag);
                rfid.TagLost += new TagEventHandler(rfid_TagLost);
                rfid.open();

                //Wait for a Phidget RFID to be attached before doing anything with
                //the object
                Console.WriteLine("waiting for attachment...");
                rfid.waitForAttachment();

                //turn on the antenna and the led to show everything is working
                rfid.Antenna = true;
                rfid.LED = true;

                //keep waiting and outputting events until keyboard input is entered
                Console.WriteLine("Press any key to end...");
                Console.Read();

                //turn off the led
                rfid.LED = false;

                //close the phidget and dispose of the object
                rfid.close();
            }
            catch { }
        }
    }
}
```



```

        rfid = null;
        Console.WriteLine("ok");
    }
    catch (PhidgetException ex)
    {
        Console.WriteLine(ex.Description);
    }
}

//attach event handler...display the serial number of the attached RFID phidget
static void rfid_Attach(object sender, AttachEventArgs e)
{
    Console.WriteLine("RFIDReader {0} attached!",
        e.Device.SerialNumber.ToString());
}

//detach event handler...display the serial number of the detached RFID phidget
static void rfid_Detach(object sender, DetachEventArgs e)
{
    Console.WriteLine("RFID reader {0} detached!",
        e.Device.SerialNumber.ToString());
}

//Error event handler...display the error description string
static void rfid_Error(object sender, ErrorEventArgs e)
{
    Console.WriteLine(e.Description);
}

//Print the tag code of the scanned tag
static void rfid_Tag(object sender, TagEventArgs e)
{
    Console.WriteLine("Tag {0} scanned", e.Tag);
}

//print the tag code for the tag that was just lost
static void rfid_TagLost(object sender, TagEventArgs e)
{
    Console.WriteLine("Tag {0} lost", e.Tag);
}
}
}

```

Technical Section



RFID

RFID (radio frequency identification) systems use data strings stored inside RFID tags (or transponders) to uniquely identify people or objects when they are scanned by an RFID reader. These types of systems are found in many applications such as passport protection, animal identification, inventory control systems, and secure access control systems.

RFID Protocols

In order for an RFID reader like the PhidgetRFID to communicate with an RFID tag, they must share a common protocol. This protocol acts as a set of rules for the way data is transmitted wirelessly between the reader and tag. The PhidgetRFID (as well as RFID tags sold by Phidgets) uses the EM4102 protocol. Any other tags that also use the EM4102 protocol can be used with the PhidgetRFID.

Communication and Effectiveness

RFID tags come in two main varieties: passive and active. Active tags have their own power supply which they use to power an antenna to communicate and transmit data. Passive tags derive the power they require to operate directly from the RF output of the RFID reader, and no other power supply is necessary. This makes passive tags cheaper to produce and easier to implement.

Because passive tags require a strong RF field to operate, their effective range is limited to an area in close proximity to the RFID reader. In the case of the PhidgetRFID, tags brought within approximately 3" of the reader can be read. The distance over which the RFID tag is usable is affected by such things as the tag shape and size, materials being used in the area near the reader, and the orientation of the reader and tag in respect to each other and in their operating environment. The smaller a tag, the closer it must be to the reader to operate.



Multiple Readers

Multiple PhidgetRFIDs within 1 to 2 meters may interfere with each other. This can be overcome in software by enabling the antennae of individual PhidgetRFID readers in sequence.

Starting with all readers disabled, enable the antenna of the first PhidgetRFID reader.

Wait for 100ms or more to detect any tags.

Disable the antenna of the first reader and enable the antenna of the second, and perform another wait cycle.

Multiple Tags

The PhidgetRFID offers no capability for collision detection or collision avoidance. If two tags are brought within the read field of a PhidgetRFID reader at the same time, neither tag will be read. An RFID tag should be removed from the read field before a second tag is introduced.

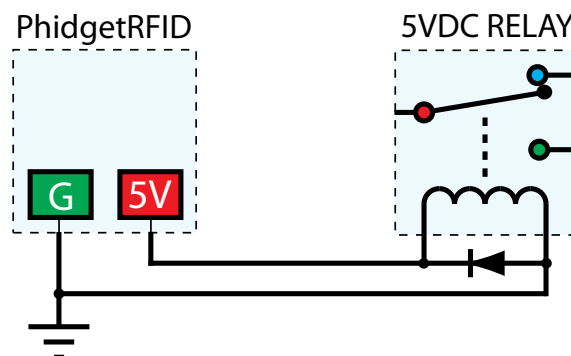
Controlled Outputs

The PhidgetRFID has four outputs - two of which are available to the user, and two of which are for internal control of the Phidget board only.

Output 0 is a +5V source from the USB bus through a P-Channel MOSFET with less than one ohm impedance. This can be used to switch a TTL or CMOS device, or it can be used to drive a 5VDC relay such as the Aromat JS1-5V. Output 1 is an LED drive output at 5VDC with maximum 15mA of available current (250 ohm CMOS output). Both Output 0 and 1 are available in hardware at the terminal blocks on the PhidgetRFID board. If Output 0 is used to drive a relay, a fast clamping diode must be placed across the relay drive pins as shown in the diagram below. Not doing so can result in permanent damage to the PhidgetRFID board.

Output	Function	Connection
0	+5VDC Source	Terminal Block
1	External LED Drive	Terminal Block
LED	Internal LED Drive	Internal Only
RF Enable	RF Antenna Enable	Internal Only

The PhidgetRFID comes equipped with an on-board LED that can be controlled using the LED property in software. Additionally, the on-board antenna can be enabled or disabled in software by using the RF Enable property. The antenna must be enabled for the PhidgetRFID to detect and read an RFID tag.



RFID Tags

The PhidgetRFID can be used with any RFID tag designed for the EM4102 protocol. RFID tags come in a variety of shapes and sizes to suit various applications. All RFID tags sold by Phidgets are guaranteed to be unique, and are available as:

- 30mm Disc Tags (these can be sewn into garments, attached to objects)
- Credit Card Sized Tags (good for security identification applications)
- Key Fob Tags (attach easily to key rings)

Visit www.phidgets.com for more information.

API (Software Technical)

We document API Calls specific to the 1023. Functions common to all Phidgets are not covered here. This section is deliberately generic - for calling conventions in a specific language, refer to that languages' API manual.

Functions

int OutputCount () [get] : Constant

Returns the number of digital outputs available on this PhidgetRFID. These are the outputs provided by the terminal block.

bool OutputState (int OutputIndex) [get,set]

Sets/Returns the state of an output. True indicates activated, False deactivated. False is the default state.

bool AntennaOn() [get,set]

Sets/Returns the state of the antenna. True turns the antenna on, False turns it off. The antenna is by default turned off, and needs to be explicitly activated before tags can be read.

bool LEDOn() [get,set]

Sets/Returns the state of the onboard LED. True turns the LED on, False turns it off. The LED is by default turned off.

string LastTag () [get]

Returns the last tag read. This method will only return a valid tag after a tag has been seen. This method can be used even after a tag has been removed from range of the reader

bool TagStatus() [get]

Returns the state of whether or not a tag is being read by the reader. True indicates that a tag is on (or near) the reader.

Events

OnOutputChange(int OutputIndex, bool State) [event]

An event issued when an output has changed.

OnTag(string) [event]

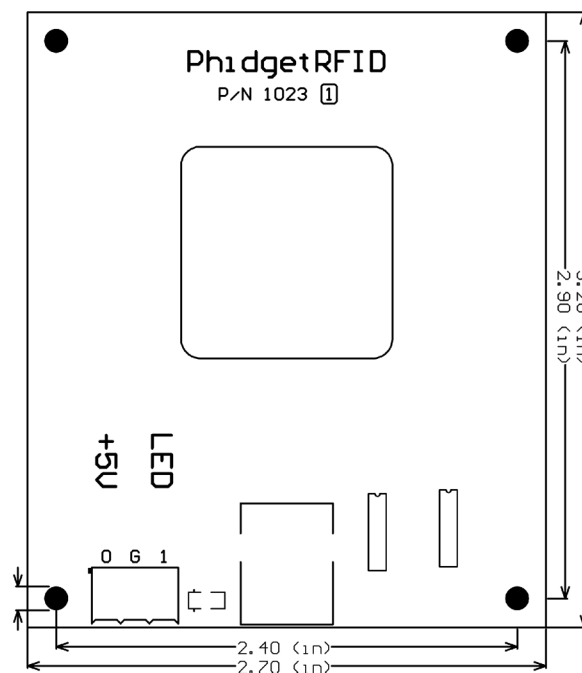
An event issued when a new tag is seen by the reader. The event is only fired one time for a new tag, so the tag has to be removed and then replaced before another OnTag event will fire.

OnTagLost(string) [event]

An event issued when a tag is removed from the reader.

Mechanical Drawing

1:1 scale



Device Specifications

Antenna Output Power (max, far field)	< 10 μ W
Antenna Resonant Frequency	125kHz - 140kHz
Communication Protocol	EM4102
Read Update Rate	30 updates / second
External +5V Supply Voltage	5VDC
External +5V Supply Current Limit	400mA
External LED Supply Voltage	5VDC
External LED Supply Current Limit	16mA
External LED Output Resistance	250 Ohms
Recommended Terminal Wire Size	16 - 26 AWG
Terminal Wire Strip Length	5 - 6mm (0.196" - 0.236")
USB-Power Current Specification	500mA max
Device Quiescent Current Consumption	16mA
Device Active Current Consumption	100mA max
Typical Read Distance - Credit Card Tag	11cm (5")
Typical Read Distance - Disk Tag	6cm (3")
Typical Read Distance - Key Fob Tag	7cm (3.5")

Product History

Date	Product Revision	Comment
June 2002	DeviceVersion 100	Product Release.
April 2004	DeviceVersion 200	Onboard LED, 2 Digital Outputs added. Ability to enable/disable Antenna added.
Jan 2005	DeviceVersion 201	RF Circuitry upgraded to improve reliability.
Jan 2006	DeviceVersion 202	Onboard microprocessor upgraded to Flash version.
Jun 2006	DeviceVersion 204	Low Voltage Reset set at 4.7 Volts
April 2007	PCB Rev 0, DeviceVersion 205	Protocol parsing bug fixed which garbled top 3 bits of RFID Tag.
July 2007	PCB Rev 1, DeviceVersion 206	Unused internal I/O pulled high out of an abundance of caution, bus current characterized. Terminal block moved to edge of PCB, center of antenna routed out, digital output transients on startup eliminated.

